SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY
HUNASAMARANAHALLI, BENGALURU 562157

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Project Entitled

"VERTICAL BLOCK BREAKER GAME"

Submitted by

KARAN SAXENA	**1MV13CS047**
KUMAR SAURAV	**1MV13CS052**

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi – 590018, Karnataka State, India

Mini Project Entitled

"Vertical Block Breaker Game"

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

For the academic year 2015-2016

Submitted by:

Karan Saxena **1MV13CS047**

Kumar Saurav **1MV13CS052**

Carried out at

Sir M. Visvesvaraya Institute of Technology

Bangalore - 562157

Under Guidance of

Mr. Elaiyaraja P.

Asst. Professor

SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BENGALURU – 562157

SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belgavi)

Bangalore – 562157

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"VERTICAL BLOCK BREAKER GAME"** is a bonafide work carried by **KARAN SAXENA (1MV13CS047)** and **KUMAR SAURAV (1MV13CS052)** in partial fulfillment for the degree of **Bachelor of Engineering** in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belgaum** during the **academic year 2015 – 2016** in **Computer Graphics and Visualization Laboratory.** The project has been hereby approved as it satisfies the academic requirements in respect of the project work prescribed for the course of Bachelor of Engineering Degree.

……	……
Signature of the guide	**Signature of the HOD**
Mr. Elaiyaraja P.	**Prof. Dilip K. Sen**
Asst. professor, Dept. of CSE	**HOD, Dept. of CSE**

Examiners:

1. ……

2. ……

ACKNOWLEDGEMENT

The successful completion of any work depends upon the cooperation and help of many people and not just those who directly execute the work. It is difficult to express in words our profound sense of gratitude to those who helped us, but we make a humble attempt to do so.

I express my sincere gratitude to our **Principal Dr. M.S.Indira**, Sir MVIT for providing facilities.

I wish to place on record my sincere thanks to **Prof Dilip K. Sen, Head of the Department**, Computer Science and Engineering for encouragement and support.

We express our sincere gratitude to our internal guide **Mr. Ilaiyaraja P., Asst. Professor in Computer Science and Engineering Department** for giving us the valuable suggestions regarding the project.

We also thank the members of the faculty of CSE department and members of Coldplay, whose suggestions helped us in the course of this project.

Our heartfelt thanks to all the above mentioned people who have contributed in the accomplishment of this project.

KARAN SAXENA (1MV13CS047)

KUMAR SAURAV (1MV13CS052)

ABSTRACT

This project demonstrates a simple game of Block Breaking. Blocks drop down vertically from top. The player has to shoot them before they touch the gun or touch the bottom of the screen. The project is keyboard interactive and we use L-R keys to perform movement of the gun.

The game will be played as follows:

- Left keyboard key to move the gun to the left.
- Right keyboard key to move the gun to the right.
- Top right close button to close the game window.
- Y keyboard to start next game.
- N keyboard key to quit the game.
- Q keyboard key force exits the game.
- Current score is displayed on the top left.
- High score is displayed on the top right.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

The term “Computer Graphics” includes almost everything on computers that is not text or sound. Today nearly all computers use some graphics and users expect to control their computer through icons and pictures rather than just by typing. Computer graphics is the field of visual computing, where one utilizes computers both to generate visual images synthetically and to integrate or alter visual and spatial information sampled from the real world. The term computer graphics has several meanings:

- The representation and manipulation of pictorial data by a computer.
- The various technologies used to create and manipulate such pictorial data.
- The sub-field of computer science which studies for digitally synthesizing and manipulating visual content.

This field can be divided into several areas: real-time 3D rendering (often used in video games), video capture and video creation rendering, special effects editing (often used for movies and television), image editing and modelling (often used for engineering and medical purposes).

1.2 History

The phrase “Computer Graphics” was coined in 1960 by William fetter, a graphic designer for Boeing. The field of computer graphics developed with the emergence of computer graphics hardware. Early projects like the whirlwind and SAGE Projects introduced the CRT as a viable display and interaction interface and introduced the light pen as an input device.

In 1959, the TX-2 computer was developed at MIT’s Lincoln Laboratory. The TX-2 integrated a number of new man-machine interfaces. A light pen could be used to draw sketches on the computer using Ivan Sutherland’s revolutionary Sketchpad software. Using a light pen, Sketchpad allowed one to draw simple shapes on computer screen, save them and even recall them later.

Many of the most important early breakthroughs in computer graphics research occurred at the University of Utah in the 1970s. A student by the name of Edwin Catmull saw

computers as the natural progression of animation and they wanted to be part of the evolution. He created an animation of his hand opening and closing. The first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden surface algorithm.

In the 1980s, artist and graphics designer began to see the personal computer, particularly the Commodore Amiga and Macintosh, as a serious design tool, one that could save time and draw more accurately than other methods. In the late 1980s, SGI computers were used to create some of the first fully computer-generated short films at Pixar.

3D graphics became more popular in the 1990s in gaming, multimedia and animation. In 1995, Toy Story, the first full-length computer generated animation film, was released in cinemas worldwide.

1.3 Applications Of Computer Graphics

- Computational biology
- Computational physics
- Computer-aided design
- Computer simulation
- Digital art
- Graphic design
- Info graphics
- Information visualization
- Scientific visualization

CHAPTER 2

COMPUTER GRAPHICS

2.1 OpenGL (Open Graphics Library):

OpenGL has become a widely accepted standard for developing graphics application. OpenGL is easy to learn, and it possesses most of the characteristics of other popular graphics system. It is top-down approach. OpenGL is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives.

OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms.

The interface between the application program and the graphics system can be specified through that set of function that resides in graphics library. The specification is called the APPLICATION PROGRAM INTERFACE (API). The application program sees only the API and is thus shielded from the details both the hardware and software implementation of graphics library. The software driver is responsible for interpreting the output of an API and converting these data to a form that is understood by the particular hardware.

Most of our applications will be designed to access openGL directly through functions in three libraries. Function in the main GL library have name that begin with the letter gl and stored in the library. The second is the openGL utility Library (GLU). This library uses only GL function but contains codes for creating common object and viewing. Rather than using a different library for each system we used available library called openGL utility toolkit (GLUT). It used as #include <glut.h>

A graphics editor is a computer program that allows users to compose and edit pictures interactively on the computer screen and save them in one of many popular “bitmap” or “raster” a format such as TIFF, JPEG, PNG and GIF.

Graphics Editors can normally be classified as:

- 2D Graphics Editors.
- 3D Graphics Editors.

A 3D Graphics Editor is used to draw 3D primitives Rectangles, Circle, polygons, etc. and alter those with operations like cut, copy, paste. These may also contain features like layers and object precision etc.

3D Graphics Editor should include the following features:

- Facilities: Cursor Movement, Editing picture objects.
- Good User Interface: GUI / Toolbars / Icon based User Interface.

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. A particular graphics software system called OpenGL, which has become a widely accepted standard for developing graphics applications.

The applications of computer graphics in some of the major areas are as follows

1. Display of information.
2. Design.
3. Simulation and Animation.
4. User interfaces.

My project titled “VERTICAL BLOCK BREAKER GAME” uses OpenGL software interface and develops 2D images. This project uses the techniques like Translation, motion, display list, transformation techniques, etc.

2.2 Problem Section Statement

Computer graphics is no longer a rarity. It is an integral part of all computer user interfaces, and is indispensable for visualizing 2D, 3D and higher dimensional objects. Creating 3D objects, rotations and any other manipulations are laborious process with graphics implementation using text editor. OpenGL provides more features for developing 3D objects with few lines by built in functions.

The geometric objects are the building blocks of any individual. Thereby developing, manipulating, applying any transformation, rotation, scaling on them is the major task of any image development.

Thereby we have put our tiny effort to develop 2D objects and perform different operations on them by using OpenGL utilities.

2.3 Existing System

The existing system involves computer graphics. Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. It includes the creation, storage and manipulation of models and images of objects.

These models include physical, mathematical, engineering, architectural and so on Computer graphics today is largely interactive –the user controls the contents, structure and appearance of objects and their displayed images by using input devices, such as keyboard, mouse or touch-sensitive panel on the screen. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television.

2.4 Proposed System

In proposed system, the OpenGL is a graphic software system designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using.

OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you

must build up your desired model from a small set of *geometric primitives* - points, lines, and polygons.

2.5 Objectives Of The Project

- Developing a package using computer graphics with OpenGL.
- Migration from text editor to OpenGL.
- To show that implementation of Translation is easier with OpenGL.
- Implementing certain technical concept like Translation, motion, and use of Idle Function.
- How to use Lightning effects used to produce computer animation.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 User Requirements

- Easy to understand and should be simple.
- The built-in functions should be utilized to the maximum extent.
- OpenGL library facilities should be used.

3.2 Hardware Requirements

- Intel Pentium CPU 2.66 GHZ
- 1 GB RAM
- Mouse
- Keyboard 108 standard
- Monitor resolution 800x600

3.3 Software Requirements

- OpenGL Tools
- Windows 32-Bit Operating System
- Turbo C++ on Dos/Windows platform

IDE: Microsoft Visual Studio 2010

CHAPTER 4

DESIGN AND IMPLEMENTATION

To design the 'Vertical Block Breaker Game' using the glut library, we need to understand various concepts, components and utility functions that are essential to implement/integrate the required visual (and audio) effects. Hence by using the following functions we design our projects.

4.1 Design

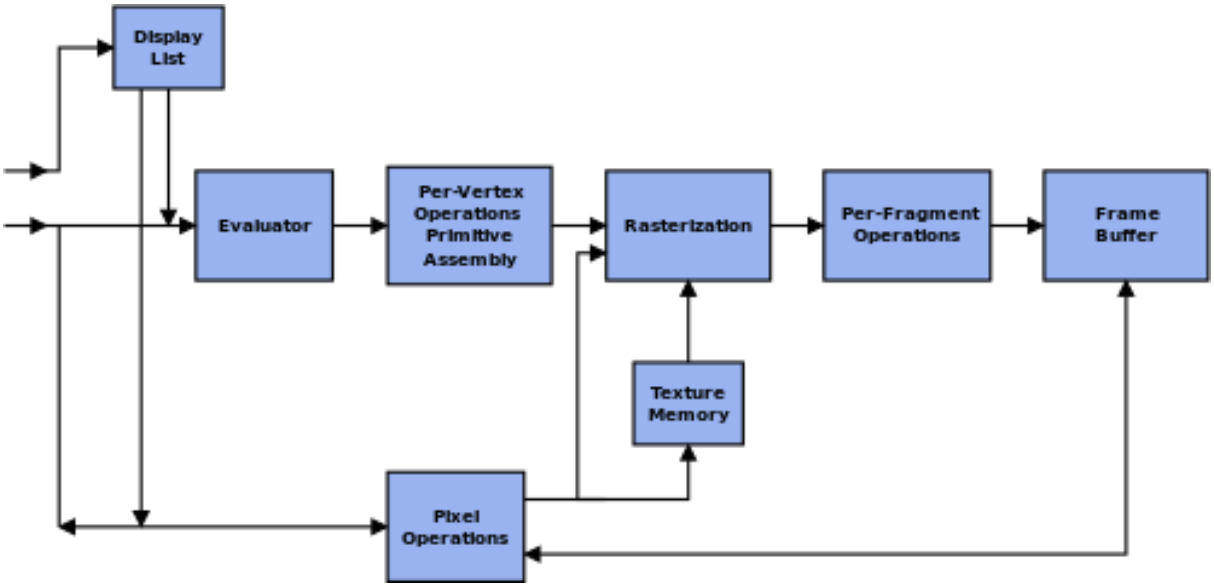

FIG 4.1 OPENGL PIPELINE

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it is possible for the API to be implemented entirely in software, it is designed to be implemented mostly or entirely in hardware.

The API is defined as a number of functions which may be called by the client program, alongside a number of named integer constants (for example, the constant GL_TEXTURE_2D, which corresponds to the decimal number 3553). Although the function definitions are superficially similar to those of the C programming language, they are language-independent. As such, OpenGL has many language bindings, some of the most noteworthy being the JavaScript binding WebGL (API, based on OpenGL ES 2.0, for 3D rendering from within a web browser); the C bindings WGL, GLX and CGL; the C binding provided by iOS; and the Java and C bindings provided by Android.

4.2 Header Files

#include <stdio.h>

stdio.h which stands for "standard input/output header" is the header in the C standard library that contains macro definitions, constants and declarations of functions and types used for various standard input and output operations.

#include <math.h>

math.h is a header file in the standard library of the C programming language designed for basic mathematical operation.

#include <windows.h>

windows.h is a Windows-specific header file for the C/C++ programming language which contains declarations for all of the functions in the Windows API. It defines a very large number of Windows specific functions that can be used in C. Here data structure used is float and integer type.

4.3 Simple Geometry

Void glBegin(glEnum mode): This function initiates a new primitive of type mode and starts the collection of vertices. Values of this mode include GL_POINTS, GL_LINE_STRIP and GL_QUADS.

Void glEnd(): Terminates a list of vertices.

glVertex2f(coordinates): This function defines the vertices of 2D figure with float as data type.

GlutInit(int argc, char *argv): Initialize GLUT. The arguments from main are passed in and can be used by the application.

glutCreateWindow(char*title): Create a Window on the display, the string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

glutDisplayMode(unsigned int mode): Request a display with the properties in mode the value of mode is determined by the logical or of options including the color model (GLUT_RGB,GLUT_INDEX) and buffering (GLUT_SINGLE,GLUT_DOUBLE).

glutInitWindowSize(intwidth,int height): Specifies the initial height and width of the window in pixels.

glutInitWindowPosition(int x, int y): Specifies the initial position of top-left corner of the window in pixels.

glutMainLoop(): Causes the program to enter an event processing loop. It should be the last statement in main.

glutDisplayFunc(void (*func)(void)): Registers the display function that is executed when the window needs to be redrawn.

glutPostRedisplay(): Requests that the display callback be executed after the current callback returns.

4.4 Interaction

glutKeyboardfunc(void(*func)(unsigned char key,int x,int y)): Sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback.

glutIdleFunc(void(*f)(void)P): Returns an identifier for a top-level menu and registers the callback function f that returns an integer value corresponding to the menu entry selected.

4.5 Transformations

glMatrixMode(GL_PROJECTION): Here the mode will be projection mode, specifies subsequent transformation matrix to an identity matrix.

glLoadIdentity(): Set the current transformation matrix to an identity stack corresponding to the current matrix mode.

4.6 Viewing.

gluOrtho2D(left,right,bottom,top): It defines a two dimensional viewing rectangle in the plane z=0.

4.7 Main Function

The main function must be called before any other GLUT/OpenGL calls. Execution of any program starts at main function. The main function gets argument c and argument v.

```
int main (int argc,char** argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
        glutInitWindowSize(600,600);
        glutInitWindowPosition(0,0);
        glutCreateWindow("brickbreaker");
        glutDisplayFunc(display);
        glutKeyboardFunc(keyb);
        glutTimerFunc(1000, bullet, 1);
        glutTimerFunc(40, updatebullet, 1);
        glutIdleFunc(idle);
        glutSpecialFunc(Keys);
        myinit();
        glutMainLoop();
        return 0;
}
```

CHAPTER 5

SOURCE CODE

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<GL/glut.h>
static GLfloat shooter [5][2]={{40,10},{80,10},{80,15},{60,19.5},{40,15}};
static GLfloat bul[10][2];
int activebulletcount;
int score = 0;
int hiscore = 0;
bool gameover;
char buffer [50];

typedef struct
{
    int x1,y1,x2,y2;
    int color;
    int lives;
}bricktype;

bricktype br[15][30];
GLfloat colorarr [][4]={{0.8,0.8,0.8},{1.0,0.0,0.0},{0.0,1.0,0.0},{0.0,0.0,1.0}};
int currbullet=6;

//Function to write characters on openGL window
void drawString (void * font, char *s, float x, float y)
{
    unsigned int i;
    glRasterPos2f(x, y);
    for (i = 0; i < strlen (s); i++)
        glutBitmapCharacter (font, s[i]);
```

```
}

void brick (int x0,int y0,int x1,int y1,int col)
{
        br[x0/40] [y0/20].x1=x0;
        br[x0/40] [y0/20].y1=y0;
        br[x0/40] [y0/20].x2=x1;
        br[x0/40] [y0/20].y2=y1;
        br[x0/40] [y0/20].color=col;
        br[x0/40] [y0/20].lives=col;
}

int brickline(int y0,int y1)
{
        int x0=0, x1=40, i;
        int x;
        for(i=0; i<15; i++)
                {
                x=1+rand()%3;
                                brick(x0,y0,x1,y1,x);
                                x0=x0+40;
                                x1=x1+40;

                }

        return 1;
}

void translatedown()
{
int x0,y0;

    for(x0=0;x0<=560;x0+=40)
    {
```

```
            if(br[x0/40][2].lives > 0 && br[x0/40][2].color != 0)
                gameover = true;
        }


for(x0=0;x0<=560;x0+=40)
    for(y0=20;y0<=580;y0+=20)
    {
        br[x0/40][(y0-20)/20].x1=br[x0/40][y0/20].x1;
        br[x0/40][(y0-20)/20].y1=br[x0/40][y0/20].y1-20;
        br[x0/40][(y0-20)/20].x2=br[x0/40][y0/20].x2;
        br[x0/40][(y0-20)/20].y2=br[x0/40][y0/20].y2-20;
        br[x0/40][(y0-20)/20].color=br[x0/40][y0/20].color;
        br[x0/40][(y0-20)/20].lives=br[x0/40][y0/20].lives;
    }
}

static int y0=580,y1=560;

void updatebullet(int value)
{
    int i;
    glutTimerFunc(40, updatebullet, 0);
    if(gameover == false)
    {
        if(activebulletcount < 10)
        {
            bul[activebulletcount][0]=shooter[3][0];
                bul[activebulletcount][1]=shooter[3][1] - 20;
                activebulletcount = activebulletcount + 1;
        }
        for(i=0;i<activebulletcount;i++)
        {
           bul[i][1]+=20;
```

```
            if(bul[i][1] > 600)
            {
                bul[i][0]=shooter[3][0];
                bul[i][1]=shooter[3][1];
            }
        }
    }
}

void bullet(int value)
{
    if(gameover == false)
    {
currbullet+=1;
        currbullet=currbullet%20;
        if(currbullet%2==0)
        {
        translatedown();
        brickline(y0,y1);
        }
    }
        glutTimerFunc(1000,bullet,1);
}

void idle()
{
int i;
  for(i=0;i<10;i++)
  {
      if((bul[i][1]<600)&&(br[((int)bul[i][0])/40][((int)bul[i][1])/20 + 1].lives>0))
      {
          score = score + 100;
          if(score > hiscore)
              hiscore = score;
```

```
                br[((int)bul[i][0])/40][((int)bul[i][1])/20 + 1].lives-=1;
                        bul[i][0]=shooter[3][0];
                         bul[i][1]=shooter[3][1];
          }
        }
    glutPostRedisplay();
}

void gun(){
            glColor3f(1.0,0.6,0.1);
            glBegin(GL_POLYGON);
            glVertex2f(shooter[0][0],shooter[0][1]);
            glVertex2f(shooter[1][0],shooter[1][1]);
            glVertex2f(shooter[2][0],shooter[2][1]);
            glVertex2f(shooter[3][0],shooter[3][1]);
            glVertex2f(shooter[4][0],shooter[4][1]);
            glEnd();
}

void drawbricks()
{
int x0,y0;
      for(x0=0;x0<=560;x0+=40)
            for(y0=580;y0>=0;y0-=20)
            {
                  if(br[x0/40][y0/20].lives>=1)
                  {
                  glColor3fv(colorarr[br[x0/40][y0/20].color]);
                  glBegin(GL_POLYGON);
                  glVertex2i(br[x0/40][y0/20].x1,br[x0/40][y0/20].y1);
                  glVertex2i(br[x0/40][y0/20].x2,br[x0/40][y0/20].y1);
                  glVertex2i(br[x0/40][y0/20].x2,br[x0/40][y0/20].y2);
                  glVertex2i(br[x0/40][y0/20].x1,br[x0/40][y0/20].y2);
```

```
                glEnd();
                }
                else            {
                                glColor3fv(colorarr[0]);
                                glBegin(GL_POLYGON);
                                glVertex2i(br[x0/40][y0/20].x1,br[x0/40][y0/20].y1);
                                glVertex2i(br[x0/40][y0/20].x2,br[x0/40][y0/20].y1);
                                glVertex2i(br[x0/40][y0/20].x2,br[x0/40][y0/20].y2);
                                glVertex2i(br[x0/40][y0/20].x1,br[x0/40][y0/20].y2);
                                glEnd();
                                }
        }
}

void display()
{
int i;
        glClear(GL_COLOR_BUFFER_BIT);
        drawbricks();
        for(i=0;i<activebulletcount;i++)
        {
             glColor3f(0.0,0.0,0.0);
            glBegin(GL_POINTS);
            glVertex2d(bul[i][0],bul[i][1]);
            glEnd();
        }
        gun();

         glColor3f(0, 0, 0);
        drawString(GLUT_BITMAP_HELVETICA_18, "Score:", 25, 585);
        itoa(score, buffer, 10);
        drawString(GLUT_BITMAP_HELVETICA_18, buffer, 85, 585);

        glColor3f(0, 0, 0);
```

```
        drawString(GLUT_BITMAP_HELVETICA_18, "Hi-Score:", 380, 585);
        itoa(hiscore, buffer, 10);
        drawString(GLUT_BITMAP_HELVETICA_18, buffer, 470, 585);

        if(gameover == true)
        {
            glColor3f(1, 1, 1);
            drawString(GLUT_BITMAP_HELVETICA_18, "Game Over", 250, 280);
//display Game Over on the screen
            drawString(GLUT_BITMAP_HELVETICA_18, "Do you want to
continue(y/n)", 180, 310); //display Do you want to continue(y/n) on the screen
        }

        glutSwapBuffers();
}

void resetBricks()
{
    int x0,y0;
    score = 0;
    for(x0=0;x0<=560;x0+=40)
        for(y0=580;y0>=0;y0-=20)
        {
            br[x0/40][y0/20].x1=0;
            br[x0/40][y0/20].y1=0;
            br[x0/40][y0/20].x2=0;
            br[x0/40][y0/20].y2=0;
            br[x0/40][y0/20].color=0;
            br[x0/40][y0/20].lives=0;
        }
}

void myinit()
{
```

```
glClearColor(0.8,0.8,0.8,1.0);
glMatrixMode(GL_PROJECTION);
glColor3f(1.0,0.0,0.0);
glPointSize(7.0);
gluOrtho2D(0.0,599.0,0.0,599.0);
glMatrixMode(GL_MODELVIEW);
      resetBricks();
}

void Keys(int key,int x,int y){
int j;
            if(gameover == false)
            {
                  switch(key){
                  case GLUT_KEY_LEFT:if(shooter[0][0]>=40){for(j=0;j<5;j++)
                                                      shooter[j][0]-=40;
                                                     glutPostRedisplay();
                                    }break;
                  case GLUT_KEY_RIGHT:if(shooter[0][0]<560){for(j=0;j<5;j++)
                                                            shooter[j][0]+=40;
                                                           glutPostRedisplay();
                                                         }break;
                  }
            }
}

void keyb(unsigned char key,int x,int y){
            switch(key){
            case 'Q':exit(0);
            case 'q':exit(0);
           case 'Y':
            case 'y':
                        if(gameover == true)
                        {
```

```
                    resetBricks();
                    gameover = false;
                }
                break;
        case 'N':
        case 'n':
                if(gameover == true)
                {
                    exit(0);
                }

                        break;
        case 27:
            exit(0);
        }
}

int main(int argc,char** argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
        glutInitWindowSize(600,600);
        glutInitWindowPosition(0,0);
        glutCreateWindow("brickbreaker");
        glutDisplayFunc(display);
        glutKeyboardFunc(keyb);
        glutTimerFunc(1000,bullet,1);
       glutTimerFunc(40,updatebullet,1);
        glutIdleFunc(idle);
        glutSpecialFunc(Keys);
        myinit();
        glutMainLoop();
        return 0;
}
```

CHAPTER 6

SNAPSHOTS

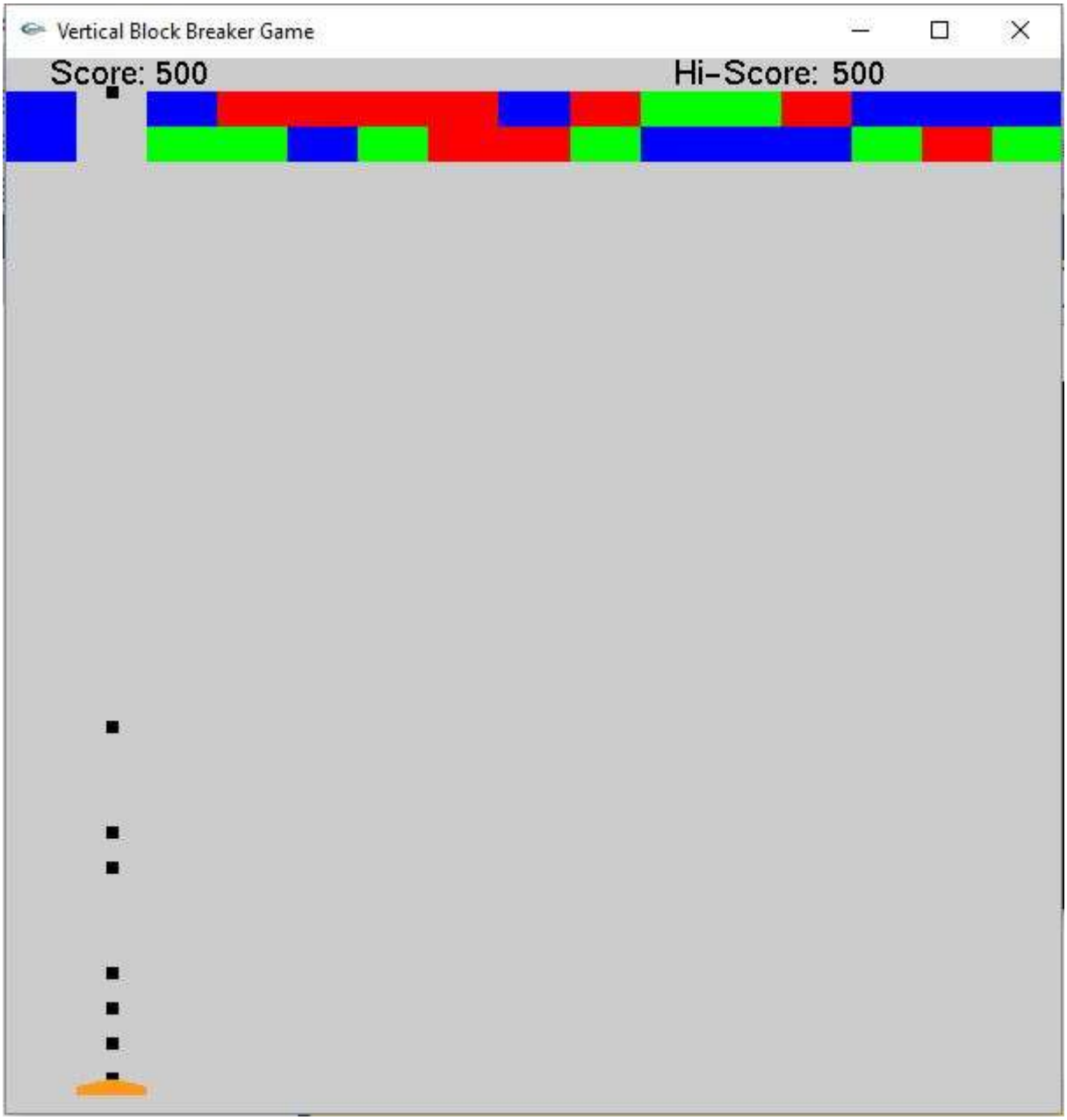

FIG 6.1 INITIAL POSITION OF THE GAME

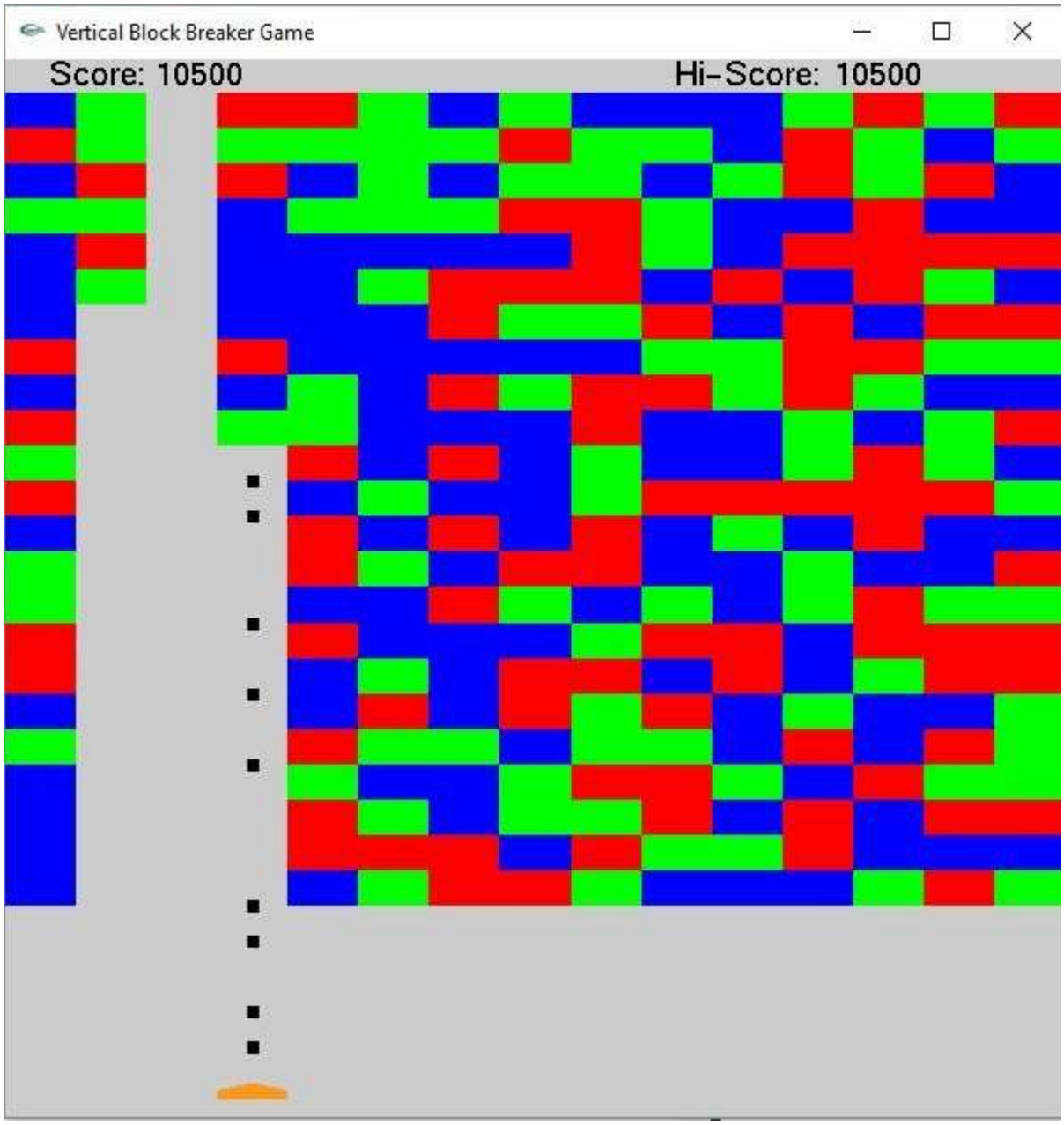

FIG 6.2 GAME IN PROGRESS

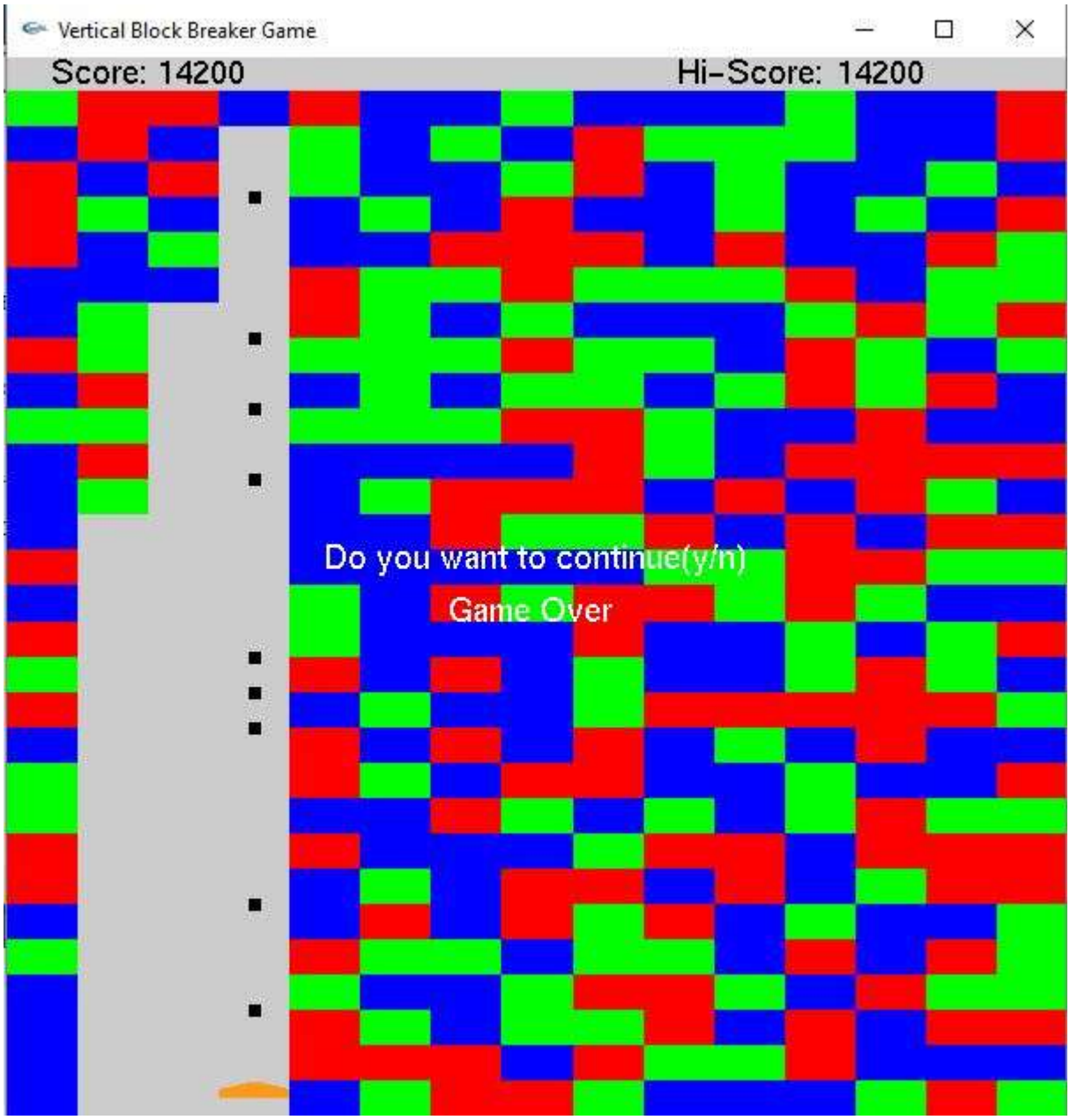

FIG 6.3 GAME OVER

CHAPTER 7

CONCLUSION

An attempt has been made to develop an OpenGL graphics package, which meets necessary requirements of the users successfully. It enables us to learn about the basic concept in OPENGL graphics and graphics and know standard library graphics function and also to explore some other function. OpenGL graphics is a huge library which consists of numerous functions.

The various shapes at lower level or to simulate any real thing animation etc. at high level. This project has given us an insight into the use of Computer graphics. As we had to use many built-in and user defined functions, we have to managed to get a certain degree of familiarity with these functions and have now understood the power of these functions. We were able to comprehend the true nature of the most powerful tool graphics in OpenGL and have understood the reason why graphics is so powerful for programmers.

We can now converse with the certain degree of confidence about graphics in openGL. Finally, we have implemented this mini projection "Vertical Block Breaker Game" using openGL package. We would like to end by saying that doing this graphics project has been a memorable experience in which we have learned a lot, although there is a scope for further improvement. We got to know a lot of different applications of OPENGL while doing this project.

FURTHER SCOPE

Scope of further improvement:

- The game can include 3D graphics.
- Various lightening effects can be implemented to make it more attractive.
- The design of the blocks can be made more attractive by using various shading techniques.

CHAPTER 8

BIBLIOGRAPHY

Books:

The books that helped us in implementing this project are as follows:

- **Computer Graphics** (OpenGL Version)

  Donald Hearn and Pauline Baker, 2nd edition, Pearson Education, 2003

- **Interactive Computer Graphics**

  Edward Angel, 5th edition, Addison Wesley, 2009

Reference Websites:

www.opengl.com

www.learnopengl.com

www.tutorialspoint.com